SQL versus NoSQL
Terminology
Why NoSQL

- Scaling
- Distributed Data Sources
- High cost of joins
- Great variation in data
- Design focuses on the query needs of the application
- RDBMS do not always match the needs of the application
- RDBMS not going away but might become less dominate
Some NoSQL Characteristics

• No predefined schema
• Limited or no support for declarative query language
• Focus on scalability, availability and performance
SQL versus NoSQL

• Transactions
• ACID Properties
  – Atomicity
  – Consistency
  – Isolation
  – Durability
SQL versus NoSQL

• BASE
  – Basically Available
  – Soft state
  – Eventually consistent
SQL versus NoSQL

• CAP
  – Consistent
    • All replicas contain the same view of the data
    • Clients always see the same view of the data
  – Available
    • System remains operational in the presence of failures
    • All clients can always read and write
  – Partition Tolerance
    • System remains operational in presence of communication failures or network partition

• Cap Theorem
  – Systems can only support 2 of 3
  – The idea is widely debated
SQL versus NoSQL

• Scalability
  – Horizontal
    • Distribute data and load over many servers
    • The servers do not share RAM or Disks
  – Vertical
    • Distribute load over many cores or processors
    • The cores or processors share RAM and Disks
SQL versus NoSQL

• Partitioning
  – Horizontal (Sharding)
    • Storing records on different servers
  – Vertical
    • Storing parts of a record on different servers

• Replication
  – Storing multiple copies of the same data
SQL versus NoSQL

- Taxonomy of NoSQL
  - Key-value
  - Document
  - Graph Database
SQL versus NoSQL

• Other terms
  – Map/reduce
  – Ad hoc Query support
  – Schema
  – Schema less
SQL versus NoSQL

• Examples
  – Redis
  – Riak
  – MongoDB
  – Hbase
  – Neo4j
MongoDB

• Document oriented database
• JSON
  – JavaScript Object Notation
  – Name/Value Pairs
  – {first:”John”, last:”Mill”}
• Documents stored in BSON format
  – Binary JSON
# MongoDB Terminology

<table>
<thead>
<tr>
<th>MongoDB Term</th>
<th>SQL Related Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Database</td>
</tr>
<tr>
<td>Collection</td>
<td>Table</td>
</tr>
<tr>
<td>Document</td>
<td>Tuple</td>
</tr>
<tr>
<td>Primary key (_id)</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Index</td>
<td>Index</td>
</tr>
</tbody>
</table>

---

**MongoDB Terminology**

- Database
- Collection
- Document
- Primary key (_id)
- Index
Mongo Shell Example

toms-air:mongodb gendreau$ mongo
MongoDB shell version: 3.2.0
connecting to: test
> use library
switched to db library
> db.author.insert( {first:"John", last:"Mill"});
WriteResult({ "nInserted" : 1 })
> db.author.findOne();
{
   "_id" : ObjectId("57238df871b1dd13fbbf38b0"),
   "first" : "John",
   "last" : "Mill"
}
Import Data From a File

toms-air:mongodb gendreau$ mongoimport --db library --collection author --type csv --headerline --file test1/author.txt
2016-04-29T11:51:31.368-0500 connected to: localhost
2016-04-29T11:51:31.370-0500 imported 19 documents
Queries

toms-air:mongodb gendreau$ mongo
MongoDB shell version: 3.2.0
connecting to: test
> db.author.find()
{ "_id" : ObjectId("57238df871b1dd13fbbf38b0"), "first" : "John", "last" : "Mill" }  
{ "_id" : ObjectId("572391135167c1034fe757a8"), "first" : "James", "last" : "Joyce" }  
{ "_id" : ObjectId("572391135167c1034fe757a9"), "first" : "Herman", "last" : "Melville" }  
{ "_id" : ObjectId("572391135167c1034fe757aa"), "first" : "Mark", "last" : "Twain" }  
...